INFORMATION GATEWAY MANAGER FOR MULTIPLE DEVICES

FIELD OF THE INVENTION

[0001] The present application relates generally to information management. In particular, the present invention is directed to an apparatus and method for managing, routing and displaying requests for information between disparate client devices and information stores.

BACKGROUND OF THE INVENTION

[0002] The proliferation of wireless devices has made obtaining information easier than ever. Many wireless "portals" exist that allow users of wireless devices (e.g., two-way pagers, cellular telephones, PDAs, etc.) to request certain types of information by sending a request to the portal. Typical types of requests include requests for driving directions, phone numbers, weather condition, stock quotes, etc. These portals are relatively simple in their operation. The portal receives the request from the wireless device, typically in the form of a specially formatted e-mail or two-way page. The portal processes the request and then returns a device-dependent formatted reply to the requestor.

[0003] Examples of such a portals are HZ.com and infobeam.net. Both allow two-way pagers and other portable devices to send special commands as an e-mail message or page to a specific address to request information from the service. However, these services are limited in that they operate on a synchronous, on-demand basis. That is, they only respond to requests in the order they are received and do not automatically forward or update requests. Also, they typically do not automatically inform users of events. Another limitation is that they are limited in the number of devices they support.

[0004] Another limitation of these systems is that they use a proprietary client interface. As such the user must keep the interface software current, otherwise, information services may not be presented. Further, these interfaces often limit the type of requests that can be made, or frustrate users because they require information the user does not have to make a request. For example, a directory service request may require that the user enter a street address of the person they wish to look-up.

[0005] There also exist so-called "push" services. These services send information without on-demand user requests. These services can be analogized to television or radio broadcasts. These services may send information such as breaking news events and the like as it occurs to anyone who subscribes to their services. However, as the name implies, these services send information without any regard for the recipient's individual needs for the information. This disadvantageously may overwhelm the user with large amounts of unwanted information and possibly create a situation where the user may miss desired information.

[0006] Yet another limitation of these systems is they are not enterprise solutions. Most corporations have large Intranets and internal data services that are not easily accessed outside the corporate infrastructure. Conventional portals do not provide a way for end users to query this internal corporate information.

[0007] Thus, there is need for a system that not only responds to on-demand requests, but also schedules and provides event driven information. Further, there is a need for a cross-platform solution that enables requests to and from many different end devices. Still further, there is a need for an information delivery system that is easily scaled and extensible to corporate infrastructures. The present invention is directed to solving the above problems, as well as other needs.

SUMMARY OF THE INVENTION

[0008] The present invention addresses the needs identified above in that it provides a distributed information processing system that includes a client device interface adapted to receive requests for information from a plurality of remote devices, a module manager adapted to receive and route said requests from said client device interface, and a plurality of information modules. The information modules register with the module manager and module manager routes the request to an appropriate one of the information modules in accordance with a type of information requested.

[0009] In accordance with a feature of the invention, the appropriate one of the plurality of information modules generates a response that is returned to the module manager, and wherein the module manager routes the response to the client interface device for delivery to a requestor.

[00010] In accordance with another feature of the invention, a subscription service maintains a subscriber database, and when the information modules send information, the subscription center is consulted to determine to which clients the information should be forwarded.

[00011] In accordance with another aspect of the invention, a method of receiving, routing and responding to requests is provided within the information processing system.

[00012] These and other aspects of the present invention will be elucidated in the following detailed description of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[00013] The foregoing summary, as well as the following detailed description of the preferred embodiments, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there is shown in the drawings an embodiment that is presently preferred, in which like references numerals represent similar parts throughout the several views of the drawings, it being understood, however, that the invention is not limited to the specific methods and instrumentalities disclosed. In the drawings:

[00014] FIG. 1 illustrates a system-level overview of the present invention;

[00015] FIG. 2. illustrates an exemplary module manager in accordance with the present invention;

[00016] FIG. 3 illustrates an exemplary process flow of user requests and responses by the information modules;

[00017] FIGS. 4 and 5 are exemplary e-mail responses formatted in HTML.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[00018] The present invention is directed to an improved system for providing information to users and for managing and routing requests from users. Referring to FIGS. 1-3, there is illustrated an exemplary system-level overview and flow diagrams of the operation of the present invention. The information delivery system 100 is preferably a platform independent application, which enables users to receive responses based upon on-demand, scheduled or event driven requests or triggers. On-demand requests are those that a user sends and, some time later, receives a response with the requested information. Scheduled requests are those provided to users who subscribe to a service offering. On a set schedule, information is sent to the user based upon the information desired and criteria selected. Event driven responses are generated based upon certain criteria or thresholds being met. For each of the above types of requests, the responses may be formatted in plain text, Hypertext Markup Language (HTML) or other format depending upon the requestor device type or the subscriber's specified e-mail client.

[00022] In the system 100 of FIG. 1, several exemplary, non-limiting requestor devices/clients are illustrated. In accordance with the present invention, the requesting device may be a pager or wireless device 110 (e.g., cellular telephone, PDA, etc.), an Internet Appliance or Thin client device 120 (e.g., special purpose e-mail device), an e-mail client 130 (e.g., Microsoft® Outlook, Eudora Mail), a Web Browser Application (e.g., Microsoft® Internet Explorer, Netscape® Navigator®, etc.) or a server 150. Any device capable of communicating electronically may operate as a client device within the system 100.

[00023] In accordance with the present invention, the clients 110-150 send a message in the format appropriate to the client device to a mailbox 200 that listens for messages (step 300). For example, the e-mail client 130 may send an HTML or plain-text formatted e-mail to the mailbox 200, whereas the server 150 may communicate directly to the mailbox 200 using an appropriate communication protocol. The mailbox 200 acts a client device interface to the system 100 and is preferably a mail server that is capable of receiving and processing Simple Mail Transport Protocol SMTP (or other) communications. The mailbox 200 may also comprise a web server or database server cable of directly receiving and queuing requests from the clients 110-150.

[00024] Upon receipt of requests (step 302), the mailbox 200 will queue and forward the requests to a Module Manager 210 (step 304). The module manager 210 mediates all interaction between the mailbox 200 and information and one or more service modules 220-260. In accordance with the invention, the Module Manager 210 is responsible for managing requests and responses to the requests.

[00025] According to the present invention, the request/response model is similar to that of the Sun Servlet API. Sun uses HttpServletResponse and HttpServletRequest as wrappers to encapsulate all the interactions between web client and web server. The present invention uses an analogous approach, allowing for enhancements to the existing Request and Response classes without recoding or even recompiling MailCenters, Module Managers, or Modules themselves. It is preferable that the requests are serializable Java objects. This allows a request to be saved to disk and restored later if the application is experiencing too high of a service load. Alternatively, a request could be encoded as

XML, placed in an HTTP header, and processed through a SOAP or XML-RPC-enabled web server.

[00026] The requests made to the module manager 210 may be either synchronous or asynchronous. Synchronous requests are handled on a first-in-first-out basis. Such requests may be directory requests, or other simple requests. Asynchronous requests may require complex or lengthy processing. Responses to these requests are sent as soon as practicable to the requester. For example, a user may request system status information for a very large network. Because this request will take time to process, the module manager 210 will recognize that subsequent user requests should not wait for this request to complete. Accordingly, the subsequent requests will be processed while the present request is processing.

[00027] The Module Manager 210 is preferably a lightweight component, where instances are created each time a new request is received and discarded after the request has been handled. This methodology reduces the wasting of CPU cycles as the module manager 210 does not run in a loop waiting for something to happen. It is also preferable that the Module Manager instances are stateless. This property enables dynamic loading of modules since each module manager 210 has no memory of what modules were available on any previous request. Such an implementation is detailed in Designing Enterprise Applications with the Java^{TM} 2 Platform, Enterprise Edition, by Nicholas Kassem and the Enterprise Team, Version 1.0.1, Final Release, October 3, 2000, which is incorporated herein by reference in its entirety. It is further preferable that each instance of the module manager 210 is multi-threaded. Accordingly, a new thread may be created for each request and discarded for the completed response.

[00028] The module manager 210 manages a master list (e.g., registry 214) of available modules, loads modules, and routes requests to the appropriate module in accordance with the requested information (step 308). The registry 214 is provided to maintain a list of registered modules within the module manager 210. The module manager 210 also provides a dynamically generated list of service handlers exported by the available modules. The list indicates what services the various registered modules support. This list may be sent to any user requesting help or who sends an undefined term as part of a request.

[00029] The module manager 210 also handles service collisions. For example, two modules could register as handlers for the same type of request (e.g., stock quotes). One method to handle this problem is an opportunistic mode wherein whichever module makes the first claim to a request owns it from then on. Another approach is to register and/or prioritize modules to determine which module receives a request.

[00030] To load modules, local and remote module loading functions 216 and 218 are provided. The loading of modules is performed dynamically and in accordance with a module's particular physical relationship to the module manager 210. Local modules may reside on the same physical device and communication between the module manager 210 and the local modules may be via memory calls, object inheritance, inter-process communication, or the like. Remote modules may be located in other devices logically connected to the module manager 210 and information is passed using other means such as via TCP/IP sockets, etc.

[00031] It is noted that dynamic loading and unloading modules allows for modules to be added to or removed from the module manager application 210 without a restart of the module manager 210. Thus, the system 100 is highly scalable and extensible to adapt to changing requirements and needs.

[00033] The Module Manager 210 routes requests to the appropriate module for processing and generating a response (step 310). For example, the module manager 210 may be aware of four registered modules, a peoplefinder module 220 (directory service), a news module 230, a markets application reliability center 240 (network applications), and a network services module 250 (server status). As noted above, other modules 260 may register with the module manger 210 to handle requests for information.

[00034] In accordance with the present invention each of the modules 220-260 presents a common interface to the Module manager 210. A common interface advantageously enables modules to be developed quickly via a software development kit (SDK). The modules 220-260 are preferably JAVA-based, however, the modules may be based in any language capable of reconstituting a request using serializable objects.

[00035] After completing its processing, the module then returns the response to the module manager 210 (step 314). The responses are preferably serializable Java objects. The module manager 210 then sends the response to the mailbox (step 316), which formats an e-mail appropriate for the requesting client device 110-140 (e.g., plain text, rich text, page, etc.). Examples of a formatted response are show in FIGS. 4 and 5, which are HTML e-mail responses sent to the e-mail application 130.

[00036] In accordance with the present invention, asynchronous subscription information may be sent to the clients 110-150. Each client or a user of a client may subscribe to receive a particular module's information (via a request or other user interface). A subscription center 270 is provided to maintain a database of subscribers to a particular module's information. When information is sent by the modules (at step 310) the subscription center 270 is consulted (step 312) to determine to which clients the information should be forwarded. Once the clients are determined at step 312, the information is forwarded as a response, as detailed above. A further description of the subscription process is explained in Scenario 2 below.

[00037] Referring again to FIG. 2, the Module Manager 210 includes a user interface (UI) 212, e.g., a graphical user interface, such that an administrator may configure the module manager 210. It is preferable that the UI allows for web-based loading and unloading of modules. This allows for the easy update of active modules and the unloading of modules that are no longer in use. Other parameters may also be configurable by the administrator using the UI 212.

[00038] Although not shown in the FIGS., the Module Manager 210 may also work with multiple mailboxes 200 at the same time. The Module Manager 210 may manage different module lists and other preferences for each Mailbox 200, or one master list in order to route requests and responses.

[00039] Usage Scenarios

[00041] Scenario 1

[00042] A vacationing user needs to reach a customer while driving her car to discuss status of a project. The user has an interactive pager 110. In accordance with the present invention, the user sends a page to the e-mail account associated with the mailbox 200. The subject line of the email shows the service being requested as well as the request parameter(s) specific to her needs. For example, the user may send the following: subject: Find S. Thurston

[00043] This request will be sent to the mailbox 200 and forwarded to the module manager 210. The module manager 210 will forward this request to the peoplefinder module 220. The peoplefinder module 220 will query a directory and return all occurrences where a person's name begins with 's' and has 'thurston' in the last name field. Each matching record may have the person's full name, phone number , interactive pager id, email address, etc.

[00044] Within minutes of sending her request, her pager vibrates with a response from the system 100 showing her all matching records. By scrolling down she can find the appropriate record so she may call e.g., Steve Thurston via her cell phone.

[00046] Scenario 2

[00047] A company employee would like to receive a listing of news and information via email so he can stay current with the corporate activities. In this scenario, the employee registers via, e.g., a corporate web site, which shows all of the news offered for subscription. During the registration process he is asked what his preferred method of correspondence is, e.g., e-mail, pager, or web-based delivery. In this scenario, he indicates e-mail and provides an e-mail address.

[00048] After viewing the subscription options, he decides to select daily clips. The subscription center 270 notes this and, at a specific time each day, sends an e-mail to the company employee listing the header of each daily clip article as well as a hyperlink to the full body of the text. Optionally, the service could send the full article.

[00049] Here, the employee is using Microsoft® Outlook® as his e-mail client 130. Accordingly, the daily clips e-mail is formatted with rich text using HTML for easy reading and a professional appearance.

[00050] Scenario 3

[00051] While in a meeting with a Business President, the subject of network and system outages comes up. The IT team is caught off guard and quickly needs to find out the number of outages in this area. Here an IT manager sends a request to the mailbox 200 asking for all outages from the past month. The module manager 210 routes this to module 240, wherein the information is determined and a response is generated that there have been 12 outages during this timeframe with an average outage duration of 2 hrs 10 minutes. The longest outage was for 3 hours 14 minutes. The IT manager may then report this information in the meeting without leaving the meeting or an extended period of delay.

[00052] Scenario 4

[00053] A server administer is at lunch when he receives a message from his manager stating that one of the servers he is responsible is down. Instead of driving to the office, he sends a message via his pager to the e-mail account associated with the mailbox 200 asking if the server is available. The module manager 210 sends the request to the network services module 250, and within a few minutes he receives a response stating that

the server is available. To further confirm the server is available, he sends another request asking if the application component of the server is operating. In just a couple of more minutes, he receives a response from the network services module 250 indicating that it was able to confirm that the application layer on the server was up and running. Finally, he sends a page to his manager that the server and application are up and running.

[00054] Various modifications of the invention, in addition to those described herein, will be apparent to those of skill in the art in view of the foregoing description. Such modifications are also intended to fall within the scope of the appended claims.